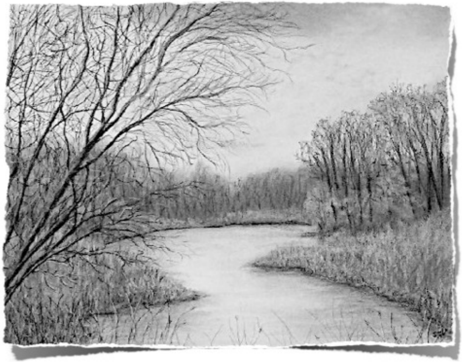
Data Streams & Communication Complexity

Lecture 1: Simple Stream Statistics in Small Space

Andrew McGregor, UMass Amherst



Data Stream Model

- *Stream:* m elements from universe of size n, e.g.,

$$\langle x_1, x_2, \ldots, x_m \rangle \quad = \quad \langle 3, 5, 3, 7, 5, 4, \ldots \rangle$$

Data Stream Model

- *Stream:* m elements from universe of size n, e.g.,

$$\langle x_1, x_2, \ldots, x_m \rangle \quad = \quad \langle 3, 5, 3, 7, 5, 4, \ldots \rangle$$

- *Goal:* Compute some function of stream, e.g., number of distinct elements, frequent items, longest increasing sequence, a clustering, graph connectivity properties, ...

Data Stream Model

- *Stream:* m elements from universe of size n, e.g.,

$$\langle x_1, x_2, \ldots, x_m \rangle = \langle 3, 5, 3, 7, 5, 4, \ldots \rangle$$

- *Goal:* Compute some function of stream, e.g., number of distinct elements, frequent items, longest increasing sequence, a clustering, graph connectivity properties, ...
- *Catch:*
  1. Limited working memory, sublinear in n and m

Data Stream Model

- *Stream:* m elements from universe of size n, e.g.,

$$\langle x_1, x_2, \ldots, x_m \rangle \quad = \quad \langle 3, 5, 3, 7, 5, 4, \ldots \rangle$$

- *Goal:* Compute some function of stream, e.g., number of distinct elements, frequent items, longest increasing sequence, a clustering, graph connectivity properties, ...
- *Catch:*
    1. Limited working memory, sublinear in n and m
    2. Access data sequentially

Data Stream Model

- *Stream:* m elements from universe of size n, e.g.,

$$\langle x_1, x_2, \ldots, x_m \rangle \quad = \quad \langle 3, 5, 3, 7, 5, 4, \ldots \rangle$$

- *Goal:* Compute some function of stream, e.g., number of distinct elements, frequent items, longest increasing sequence, a clustering, graph connectivity properties, ...
- *Catch:*
  1. Limited working memory, sublinear in n and m
  2. Access data sequentially
  3. Process each element quickly

Data Stream Model

- *Stream:* m elements from universe of size n, e.g.,

$$\langle x_1, x_2, \ldots, x_m \rangle \quad = \quad \langle 3, 5, 3, 7, 5, 4, \ldots \rangle$$

- *Goal:* Compute some function of stream, e.g., number of distinct elements, frequent items, longest increasing sequence, a clustering, graph connectivity properties, ...
- *Catch:*
  1. Limited working memory, sublinear in n and m
  2. Access data sequentially
  3. Process each element quickly
- Origins in seventies but has become popular in last ten years...

Why's it become popular?

- *Practical Appeal:*
  - Faster networks, cheaper data storage, ubiquitous data-logging results in massive amount of data to be processed.
  - Applications to network monitoring, query planning, I/O efficiency for massive data, sensor networks aggregation...

Why's it become popular?

- *Practical Appeal:*
  - Faster networks, cheaper data storage, ubiquitous data-logging results in massive amount of data to be processed.
  - Applications to network monitoring, query planning, I/O efficiency for massive data, sensor networks aggregation...
- *Theoretical Appeal:*
  - Easy to state problems but hard to solve.
  - Links to communication complexity, compressed sensing, metric embeddings, pseudo-random generators, approximation...

This Lecture: Basic Numerical Statistics

- Given a stream of m elements from universe $[n] = \{1, 2, \ldots, n\}$, e.g.,

$$\langle x_1, x_2, \ldots, x_m \rangle \quad = \quad \langle 3, 5, 3, 7, 5, 4, \ldots \rangle$$

let $f \in \mathbb{N}^n$ be the *frequency vector* where f_i is the frequency of i.

This Lecture: Basic Numerical Statistics

- Given a stream of m elements from universe $[n] = \{1, 2, \ldots, n\}$, e.g.,

$$\langle x_1, x_2, \ldots, x_m \rangle \quad = \quad \langle 3, 5, 3, 7, 5, 4, \ldots \rangle$$

  let $f \in \mathbb{N}^n$ be the *frequency vector* where f_i is the frequency of i.
- *Problems:* What can we approximate in sub linear space?
  - Frequency moments: $F_k = \sum_i f_i^k$.
  - Max frequency: $F_\infty = \max_i f_i$.
  - Number of distinct element: $F_0 = \sum_i f_i^0$
  - Median: j such that $f_1 + f_2 + \ldots + f_j \approx m/2$

  Algorithms are often randomized and guarantees will be probabilistic.

This Lecture: Basic Numerical Statistics

- Given a stream of m elements from universe $[n] = \{1, 2, \ldots, n\}$, e.g.,

$$\langle x_1, x_2, \ldots, x_m \rangle \quad = \quad \langle 3, 5, 3, 7, 5, 4, \ldots \rangle$$

  let $f \in \mathbb{N}^n$ be the *frequency vector* where f_i is the frequency of i.
- *Problems:* What can we approximate in sub linear space?
  - Frequency moments: $F_k = \sum_i f_i^k$.
  - Max frequency: $F_\infty = \max_i f_i$.
  - Number of distinct element: $F_0 = \sum_i f_i^0$
  - Median: j such that $f_1 + f_2 + \ldots + f_j \approx m/2$

  Algorithms are often randomized and guarantees will be probabilistic.
- *Keep things simple:* Could consider f_i's being increased or decreased but for this talk we'll focus on unit increments. Will also assume algorithms have an unlimited store of random bits.

Outline

Sampling

Sketching: The Basics

Count-Min and Applications

Count-Sketch: Count-Min with a Twist

ℓ_p Sampling and Frequency Moments

Sampling and Statistics

- Sampling is a general technique for tackling massive amounts of data

Sampling and Statistics

- Sampling is a general technique for tackling massive amounts of data
- *Example:* To find an *ϵ-approximate median*, i.e., j such that

$$f_1 + f_2 + \ldots + f_j = m/2 \pm \epsilon m$$

  then sampling $O(\epsilon^{-2})$ stream elements and returning the sample median works with good probability.

Sampling and Statistics

- Sampling is a general technique for tackling massive amounts of data
- *Example:* To find an *ϵ-approximate median*, i.e., j such that

$$f_1 + f_2 + \ldots + f_j = m/2 \pm \epsilon m$$

  then sampling $O(\epsilon^{-2})$ stream elements and returning the sample median works with good probability.
- *Beyond basic sampling:* There are more powerful forms of sampling and other techniques the make better use of the limited space.

AMS Sampling

- *Problem:* Estimate $\sum_i g(f_i)$ for some function g with $g(0) = 0$

AMS Sampling

- *Problem:* Estimate $\sum_i g(f_i)$ for some function g with $g(0) = 0$
- *Basic Estimator:* Sample x_J where $J \in_R [m]$ and compute

$$r = |\{j \geq J : x_j = x_J\}|$$

AMS Sampling

- *Problem:* Estimate $\sum_i g(f_i)$ for some function g with $g(0) = 0$
- *Basic Estimator:* Sample x_J where $J \in_R [m]$ and compute

$$r = |\{j \geq J : x_j = x_J\}|$$

Output $X = m(g(r) - g(r-1))$

AMS Sampling

- *Problem:* Estimate $\sum_i g(f_i)$ for some function g with $g(0) = 0$
- *Basic Estimator:* Sample x_J where $J \in_R [m]$ and compute

$$r = |\{j \geq J : x_j = x_J\}|$$

  Output $X = m(g(r) - g(r-1))$
- *Expectation:*

$$\mathbb{E}[X]$$

AMS Sampling

- *Problem:* Estimate $\sum_i g(f_i)$ for some function g with $g(0) = 0$
- *Basic Estimator:* Sample x_J where $J \in_R [m]$ and compute

$$r = |\{j \geq J : x_j = x_J\}|$$

  Output $X = m(g(r) - g(r-1))$
- *Expectation:*

$$\mathbb{E}[X] = \sum_i \mathbb{P}[x_J = i]\,\mathbb{E}[X | x_J = i]$$

AMS Sampling

- *Problem:* Estimate $\sum_i g(f_i)$ for some function g with $g(0) = 0$
- *Basic Estimator:* Sample x_J where $J \in_R [m]$ and compute

  $$r = |\{j \geq J : x_j = x_J\}|$$

  Output $X = m(g(r) - g(r-1))$
- *Expectation:*

$$\begin{aligned}
\mathbb{E}[X] &= \sum_i \mathbb{P}[x_J = i]\,\mathbb{E}[X | x_J = i] \\
&= \sum_i \frac{f_i}{m}\left(\sum_{r=1}^{f_i} \frac{m(g(r) - g(r-1))}{f_i}\right)
\end{aligned}$$

AMS Sampling

- *Problem:* Estimate $\sum_i g(f_i)$ for some function g with $g(0) = 0$
- *Basic Estimator:* Sample x_J where $J \in_R [m]$ and compute

$$r = |\{j \geq J : x_j = x_J\}|$$

  Output $X = m(g(r) - g(r-1))$
- *Expectation:*

$$\begin{aligned}
\mathbb{E}[X] &= \sum_i \mathbb{P}[x_J = i]\,\mathbb{E}[X | x_J = i] \\
&= \sum_i \frac{f_i}{m}\left(\sum_{r=1}^{f_i} \frac{m(g(r) - g(r-1))}{f_i}\right) \\
&= \sum_i g(f_i)
\end{aligned}$$

AMS Sampling

- *Problem:* Estimate $\sum_i g(f_i)$ for some function g with $g(0) = 0$
- *Basic Estimator:* Sample x_J where $J \in_R [m]$ and compute

$$r = |\{j \geq J : x_j = x_J\}|$$

  Output $X = m(g(r) - g(r-1))$
- *Expectation:*

$$\begin{aligned} \mathbb{E}[X] &= \sum_i \mathbb{P}[x_J = i] \mathbb{E}[X | x_J = i] \\ &= \sum_i \frac{f_i}{m} \left(\sum_{r=1}^{f_i} \frac{m(g(r) - g(r-1))}{f_i} \right) \\ &= \sum_i g(f_i) \end{aligned}$$

- *For high confidence:* Compute t estimators in parallel and average.

Example: Frequency Moments

- *Frequency Moments:* Define $F_k = \sum_i f_i^k$ for $k \in \{1, 2, 3, \ldots\}$

Example: Frequency Moments

- *Frequency Moments:* Define $F_k = \sum_i f_i^k$ for $k \in \{1, 2, 3, \ldots\}$
- Use AMS estimator with $X = m(r^k - (r-1)^k)$.

Example: Frequency Moments

- *Frequency Moments:* Define $F_k = \sum_i f_i^k$ for $k \in \{1, 2, 3, \ldots\}$
- Use AMS estimator with $X = m(r^k - (r-1)^k)$.
- *Expectation:* $\mathbb{E}[X] = F_k$

Example: Frequency Moments

- *Frequency Moments:* Define $F_k = \sum_i f_i^k$ for $k \in \{1, 2, 3, \ldots\}$
- Use AMS estimator with $X = m(r^k - (r-1)^k)$.
- *Expectation:* $\mathbb{E}[X] = F_k$
- *Range:* $0 \leq X \leq kmF_\infty^{k-1} \leq kn^{1-1/k}F_k$

Example: Frequency Moments

- *Frequency Moments:* Define $F_k = \sum_i f_i^k$ for $k \in \{1, 2, 3, \ldots\}$
- Use AMS estimator with $X = m(r^k - (r-1)^k)$.
- *Expectation:* $\mathbb{E}[X] = F_k$
- *Range:* $0 \leq X \leq kmF_\infty^{k-1} \leq kn^{1-1/k}F_k$
- Repeat t times and let $\tilde{F}_k$ be the average value. By Chernoff,

$$\mathbb{P}\left[|\tilde{F}_k - F_k| \geq \epsilon F_k\right] \leq 2\exp\left(-\frac{tF_k\epsilon^2}{3kn^{1-1/k}F_k}\right) = 2\exp\left(-\frac{t\epsilon^2}{3kn^{1-1/k}}\right)$$

Example: Frequency Moments

- *Frequency Moments:* Define $F_k = \sum_i f_i^k$ for $k \in \{1, 2, 3, \ldots\}$
- Use AMS estimator with $X = m(r^k - (r-1)^k)$.
- *Expectation:* $\mathbb{E}[X] = F_k$
- *Range:* $0 \leq X \leq kmF_\infty^{k-1} \leq kn^{1-1/k} F_k$
- Repeat t times and let $\tilde{F}_k$ be the average value. By Chernoff,

$$\mathbb{P}\left[|\tilde{F}_k - F_k| \geq \epsilon F_k\right] \leq 2\exp\left(-\frac{tF_k\epsilon^2}{3kn^{1-1/k}F_k}\right) = 2\exp\left(-\frac{t\epsilon^2}{3kn^{1-1/k}}\right)$$

- If $t = 3\epsilon^{-2} kn^{1-1/k} \log(2\delta^{-1})$ then $\mathbb{P}\left[|\tilde{F}_k - F_k| \geq \epsilon F_k\right] \leq \delta$.

Example: Frequency Moments

- *Frequency Moments:* Define $F_k = \sum_i f_i^k$ for $k \in \{1, 2, 3, \ldots\}$
- Use AMS estimator with $X = m(r^k - (r-1)^k)$.
- *Expectation:* $\mathbb{E}[X] = F_k$
- *Range:* $0 \le X \le kmF_\infty^{k-1} \le kn^{1-1/k}F_k$
- Repeat t times and let $\tilde{F}_k$ be the average value. By Chernoff,

$$\mathbb{P}\left[|\tilde{F}_k - F_k| \ge \epsilon F_k\right] \le 2\exp\left(-\frac{tF_k\epsilon^2}{3kn^{1-1/k}F_k}\right) = 2\exp\left(-\frac{t\epsilon^2}{3kn^{1-1/k}}\right)$$

- If $t = 3\epsilon^{-2}kn^{1-1/k}\log(2\delta^{-1})$ then $\mathbb{P}\left[|\tilde{F}_k - F_k| \ge \epsilon F_k\right] \le \delta$.
- *Thm:* In $\tilde{O}(\epsilon^{-2}n^{1-1/k})$ space we can find a $(1 \pm \epsilon)$ approximation for F_k with probability at least $1 - \delta$.

Outline

Sampling

Sketching: The Basics

Count-Min and Applications

Count-Sketch: Count-Min with a Twist

ℓ_p Sampling and Frequency Moments

Random Projections

- Many stream algorithms use a random projection $Z \in \mathbb{R}^{w \times n}$, $w \ll n$

$$Z(f) = \begin{bmatrix} z_{1,1} & \dots & & \dots & z_{1,n} \\ \vdots & & & & \vdots \\ z_{w,1} & \dots & & \dots & z_{w,n} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ \\ \vdots \\ f_n \end{bmatrix} = \begin{bmatrix} s_1 \\ \vdots \\ s_w \end{bmatrix} = s$$

Random Projections

- Many stream algorithms use a random projection $Z \in \mathbb{R}^{w \times n}$, $w \ll n$

$$Z(f) = \begin{bmatrix} z_{1,1} & \cdots & \cdots & z_{1,n} \\ \vdots & & & \vdots \\ z_{w,1} & \cdots & \cdots & z_{w,n} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ \\ \vdots \\ f_n \end{bmatrix} = \begin{bmatrix} s_1 \\ \vdots \\ s_w \end{bmatrix} = s$$

- *Updatable:* We can maintain sketch s in $\tilde{O}(w)$ space since incrementing f_i corresponds to

Random Projections

- Many stream algorithms use a random projection $Z \in \mathbb{R}^{w \times n}$, $w \ll n$

$$Z(f) = \begin{bmatrix} z_{1,1} & \cdots & & \cdots & z_{1,n} \\ \vdots & & & & \vdots \\ z_{w,1} & \cdots & & \cdots & z_{w,n} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ \\ \vdots \\ f_n \end{bmatrix} = \begin{bmatrix} s_1 \\ \vdots \\ s_w \end{bmatrix} = s$$

- *Updatable:* We can maintain sketch s in $\tilde{O}(w)$ space since incrementing f_i corresponds to

$$s \leftarrow s + \begin{bmatrix} z_{1,i} \\ \vdots \\ z_{w,i} \end{bmatrix}$$

Random Projections

- Many stream algorithms use a random projection $Z \in \mathbb{R}^{w \times n}$, $w \ll n$

$$Z(f) = \begin{bmatrix} z_{1,1} & \dots & \dots & z_{1,n} \\ \vdots & & & \vdots \\ z_{w,1} & \dots & \dots & z_{w,n} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ \\ \vdots \\ f_n \end{bmatrix} = \begin{bmatrix} s_1 \\ \vdots \\ s_w \end{bmatrix} = s$$

- *Updatable:* We can maintain sketch s in $\tilde{O}(w)$ space since incrementing f_i corresponds to

$$s \leftarrow s + \begin{bmatrix} z_{1,i} \\ \vdots \\ z_{w,i} \end{bmatrix}$$

- *Useful:* Choose a distribution for $z_{i,j}$ such that relevant function of f can be estimated from s with high probability for sufficiently large w.

Examples

- If $z_{i,j} \in_R \{-1, 1\}$, can estimate F_2 with $w = O(\epsilon^{-2} \log \delta^{-1})$.

Examples

- If $z_{i,j} \in_R \{-1, 1\}$, can estimate F_2 with $w = O(\epsilon^{-2} \log \delta^{-1})$.
- If $z_{i,j} \sim \mathcal{D}$ where $\mathcal{D}$ is p-stable $p \in (0, 2]$, can estimate F_p with $w = O(\epsilon^{-2} \log \delta^{-1})$. For example, 1 and 2 stable distributions are:

$$\text{Cauchy}(x) = \frac{1}{\pi} \cdot \frac{1}{1 + x^2} \quad \text{Gaussian}(x) = \frac{1}{\sqrt{2\pi}} \cdot e^{-x^2/2}$$

Examples

- If $z_{i,j} \in_R \{-1, 1\}$, can estimate F_2 with $w = O(\epsilon^{-2} \log \delta^{-1})$.
- If $z_{i,j} \sim \mathcal{D}$ where $\mathcal{D}$ is p-stable $p \in (0, 2]$, can estimate F_p with $w = O(\epsilon^{-2} \log \delta^{-1})$. For example, 1 and 2 stable distributions are:

$$\text{Cauchy}(x) = \frac{1}{\pi} \cdot \frac{1}{1 + x^2} \quad \text{Gaussian}(x) = \frac{1}{\sqrt{2\pi}} \cdot e^{-x^2/2}$$

- Note that $F_0 = (1 \pm \epsilon)F_p$ if $p = \log(1 + \epsilon)/\log m$.

Examples

- If $z_{i,j} \in_R \{-1, 1\}$, can estimate F_2 with $w = O(\epsilon^{-2} \log \delta^{-1})$.
- If $z_{i,j} \sim \mathcal{D}$ where $\mathcal{D}$ is p-stable $p \in (0, 2]$, can estimate F_p with $w = O(\epsilon^{-2} \log \delta^{-1})$. For example, 1 and 2 stable distributions are:

$$\text{Cauchy}(x) = \frac{1}{\pi} \cdot \frac{1}{1 + x^2} \quad \text{Gaussian}(x) = \frac{1}{\sqrt{2\pi}} \cdot e^{-x^2/2}$$

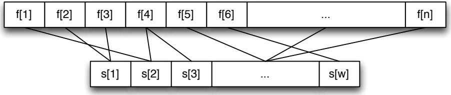
- Note that $F_0 = (1 \pm \epsilon) F_p$ if $p = \log(1 + \epsilon)/\log m$.
- For the rest of lecture we'll focus on "hash-based" sketches. Given a random hash function $h : [n] \to [w]$, non-zero entries are $z_{h_i, i}$.

$$Z = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Outline

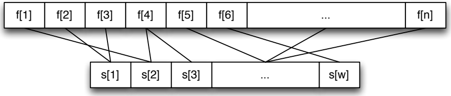
Count-Min Sketch

- Maintain vector $s \in \mathbb{N}^w$ via random hash function $h : [n] \to [w]$



Count-Min Sketch

- Maintain vector $s \in \mathbb{N}^w$ via random hash function $h : [n] \to [w]$

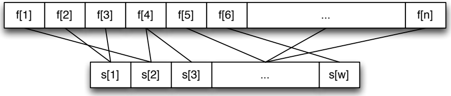


- *Update:* For each increment of f_i, increment s_{h_i}. Hence,

$$s_k = \sum_{j:h_j=k} f_j$$

Count-Min Sketch

- Maintain vector $s \in \mathbb{N}^w$ via random hash function $h : [n] \to [w]$

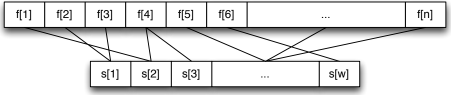


- *Update:* For each increment of f_i, increment s_{h_i}. Hence,

$$s_k = \sum_{j:h_j=k} f_j \qquad \text{e.g., } s_3 = f_6 + f_7 + f_{13}$$

Count-Min Sketch

- Maintain vector $s \in \mathbb{N}^w$ via random hash function $h : [n] \to [w]$



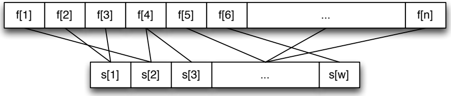
- *Update:* For each increment of f_i, increment s_{h_i}. Hence,

$$s_k = \sum_{j:h_j=k} f_j \qquad \text{e.g., } s_3 = f_6 + f_7 + f_{13}$$

- *Query:* Use $\tilde{f}_i = s_{h_i}$ to estimate f_i.

Count-Min Sketch

- Maintain vector $s \in \mathbb{N}^w$ via random hash function $h : [n] \to [w]$



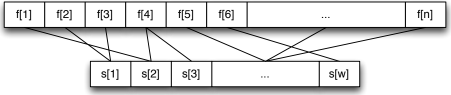
- *Update:* For each increment of f_i, increment s_{h_i}. Hence,

$$s_k = \sum_{j:h_j=k} f_j \qquad \text{e.g., } s_3 = f_6 + f_7 + f_{13}$$

- *Query:* Use $\tilde{f}_i = s_{h_i}$ to estimate f_i.
- *Lemma:* $f_i \leq \tilde{f}_i$ and $\mathbb{P}\left[\tilde{f}_i \geq f_i + 2m/w\right] \leq 1/2$

Count-Min Sketch

- Maintain vector $s \in \mathbb{N}^w$ via random hash function $h : [n] \to [w]$



- *Update:* For each increment of f_i, increment s_{h_i}. Hence,

$$s_k = \sum_{j:h_j=k} f_j \qquad \text{e.g., } s_3 = f_6 + f_7 + f_{13}$$

- *Query:* Use $\tilde{f}_i = s_{h_i}$ to estimate f_i.
- *Lemma:* $f_i \le \tilde{f}_i$ and $\mathbb{P}\left[\tilde{f}_i \ge f_i + 2m/w\right] \le 1/2$
- *Thm:* Let $w = 2/\epsilon$. Repeat the hashing $\lg(\delta^{-1})$ times in parallel and take the minimum estimate for f_i

$$\mathbb{P}\left[f_i \le \tilde{f}_i \le f_i + \epsilon m\right] \ge 1 - \delta$$

Proof of Lemma

- Define $\mathcal{E}$ by $\tilde{f}_i = f_i + \mathcal{E}$ and so

$$\mathcal{E} = \sum_{j \neq i : h_i = h_j} f_j$$

Proof of Lemma

- Define $\mathcal{E}$ by $\tilde{f}_i = f_i + \mathcal{E}$ and so

$$\mathcal{E} = \sum_{j \neq i : h_i = h_j} f_j$$

- Since all $f_j \geq 0$, we have $\mathcal{E} \geq 0$.

Proof of Lemma

- Define $\mathcal{E}$ by $\tilde{f}_i = f_i + \mathcal{E}$ and so

$$\mathcal{E} = \sum_{j \neq i : h_i = h_j} f_j$$

- Since all $f_j \geq 0$, we have $\mathcal{E} \geq 0$.
- Since $\mathbb{P}[h_i = h_j] = 1/w$,

$$\mathbb{E}[\mathcal{E}] = \sum_{j \neq i} f_j \cdot \mathbb{P}[h_i = h_j]$$

Proof of Lemma

- Define $\mathcal{E}$ by $\tilde{f}_i = f_i + \mathcal{E}$ and so

$$\mathcal{E} = \sum_{j \neq i : h_i = h_j} f_j$$

- Since all $f_j \geq 0$, we have $\mathcal{E} \geq 0$.
- Since $\mathbb{P}\left[h_i = h_j\right] = 1/w$,

$$\mathbb{E}\left[\mathcal{E}\right] = \sum_{j \neq i} f_j \cdot \mathbb{P}\left[h_i = h_j\right] \leq m/w$$

Proof of Lemma

- Define $\mathcal{E}$ by $\tilde{f}_i = f_i + \mathcal{E}$ and so

$$\mathcal{E} = \sum_{j \neq i : h_i = h_j} f_j$$

- Since all $f_j \geq 0$, we have $\mathcal{E} \geq 0$.
- Since $\mathbb{P}[h_i = h_j] = 1/w$,

$$\mathbb{E}[\mathcal{E}] = \sum_{j \neq i} f_j \cdot \mathbb{P}[h_i = h_j] \leq m/w$$

- By an application of the Markov bound,

$$\mathbb{P}[\mathcal{E} \geq 2m/w] \leq 1/2$$

Range Queries

- *Range Query:* For $i, j \in [n]$, estimate $f_{[i,j]} = f_i + f_{i+1} + \ldots + f_j$

Range Queries

- *Range Query:* For $i, j \in [n]$, estimate $f_{[i,j]} = f_i + f_{i+1} + \ldots + f_j$
- *Dyadic Intervals:* Restrict attention to intervals of the form

$$[1 + (i-1)2^j, i2^j] \quad \text{where } j \in \{0, 1, \ldots, \lg n\}, i \in \{1, 2, \ldots n/2^j\}$$

Range Queries

- *Range Query:* For $i, j \in [n]$, estimate $f_{[i,j]} = f_i + f_{i+1} + \ldots + f_j$
- *Dyadic Intervals:* Restrict attention to intervals of the form

$$[1 + (i-1)2^j, i2^j] \quad \text{where } j \in \{0, 1, \ldots, \lg n\}, i \in \{1, 2, \ldots n/2^j\}$$

since any range can be partitioned as $O(\log n)$ such intervals. E.g.,

$$[48, 106] = [48, 48] \cup [49, 64] \cup [65, 96] \cup [97, 104] \cup [105, 106]$$

Range Queries

- *Range Query:* For $i, j \in [n]$, estimate $f_{[i,j]} = f_i + f_{i+1} + \ldots + f_j$
- *Dyadic Intervals:* Restrict attention to intervals of the form

$$[1 + (i-1)2^j, i2^j] \quad \text{where } j \in \{0, 1, \ldots, \lg n\}, i \in \{1, 2, \ldots n/2^j\}$$

  since any range can be partitioned as $O(\log n)$ such intervals. E.g.,

$$[48, 106] = [48, 48] \cup [49, 64] \cup [65, 96] \cup [97, 104] \cup [105, 106]$$

- To support dyadic intervals, construct Count-Min sketches corresponding to intervals of width $1, 2, 4, 8, \ldots$

Range Queries

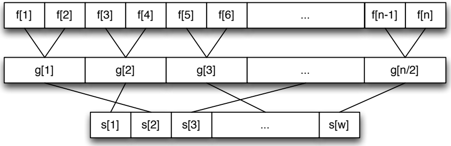
- *Range Query:* For $i, j \in [n]$, estimate $f_{[i,j]} = f_i + f_{i+1} + \ldots + f_j$
- *Dyadic Intervals:* Restrict attention to intervals of the form

$$[1 + (i-1)2^j, i2^j] \quad \text{where } j \in \{0, 1, \ldots, \lg n\}, i \in \{1, 2, \ldots n/2^j\}$$

since any range can be partitioned as $O(\log n)$ such intervals. E.g.,

$$[48, 106] = [48, 48] \cup [49, 64] \cup [65, 96] \cup [97, 104] \cup [105, 106]$$

- To support dyadic intervals, construct Count-Min sketches corresponding to intervals of width $1, 2, 4, 8, \ldots$
- E.g., for intervals of width 2 we have:



where update rule is now: for increment of f_{2i-1} or f_{2i}, increment s_{h_i}.

Quantiles and Heavy Hitters

Quantiles and Heavy Hitters

- *Quantiles:* Find j such that

$$f_1 + \ldots + f_j \approx m/2$$

Quantiles and Heavy Hitters

- *Quantiles:* Find j such that

$$f_1 + \ldots + f_j \approx m/2$$

  Can approximate median via binary search of range queries.

Quantiles and Heavy Hitters

- *Quantiles:* Find j such that

$$f_1 + \ldots + f_j \approx m/2$$

  Can approximate median via binary search of range queries.

- *Heavy Hitter Problem:* Find a set $S \subset [n]$ where

$$\{i : f_i \geq \phi m\} \subseteq S \subseteq \{i : f_i \geq (\phi - \epsilon)m\}$$

Quantiles and Heavy Hitters

- *Quantiles:* Find j such that

$$f_1 + \ldots + f_j \approx m/2$$

  Can approximate median via binary search of range queries.
- *Heavy Hitter Problem:* Find a set $S \subset [n]$ where

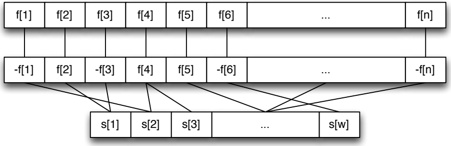
$$\{i : f_i \geq \phi m\} \subseteq S \subseteq \{i : f_i \geq (\phi - \epsilon)m\}$$

  Rather than checking each $\tilde{f}_i$ individually can save time by exploiting the fact that if $\tilde{f}_{[i,k]} < \phi m$ then $f_j < \phi m$ for all $j \in [i, k]$.

Outline

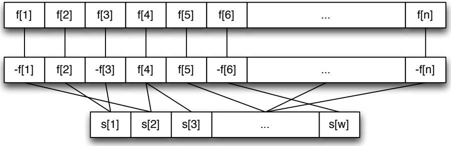
Count-Sketch: Count-Min with a Twist

- Maintain $s \in \mathbb{N}^w$ via hash functions $h : [n] \to [w]$, $r : [n] \to \{-1, 1\}$



Count-Sketch: Count-Min with a Twist

- Maintain $s \in \mathbb{N}^w$ via hash functions $h : [n] \to [w]$, $r : [n] \to \{-1, 1\}$

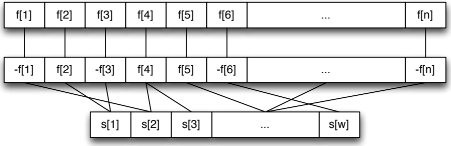


- *Update:* For each increment of f_i, $s_{h_i} \leftarrow s_{h_i} + r_i$. Hence,

$$s_k = \sum_{j:h_j=k} f_j r_j$$

Count-Sketch: Count-Min with a Twist

- Maintain $s \in \mathbb{N}^w$ via hash functions $h : [n] \to [w]$, $r : [n] \to \{-1, 1\}$

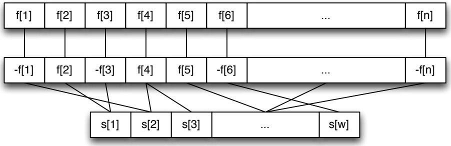


- *Update:* For each increment of f_i, $s_{h_i} \leftarrow s_{h_i} + r_i$. Hence,

$$s_k = \sum_{j: h_j = k} f_j r_j \qquad \text{e.g., } s_3 = f_6 - f_7 - f_{13}$$

Count-Sketch: Count-Min with a Twist

- Maintain $s \in \mathbb{N}^w$ via hash functions $h : [n] \to [w]$, $r : [n] \to \{-1, 1\}$



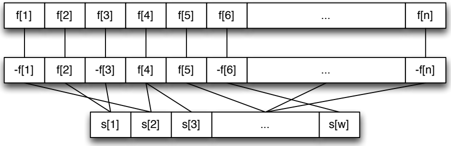
- *Update:* For each increment of f_i, $s_{h_i} \leftarrow s_{h_i} + r_i$. Hence,

$$s_k = \sum_{j:h_j=k} f_j r_j \qquad \text{e.g., } s_3 = f_6 - f_7 - f_{13}$$

- *Query:* Use $\tilde{f}_i = s_{h_i} r_i$ to estimate f_i.

Count-Sketch: Count-Min with a Twist

- Maintain $s \in \mathbb{N}^w$ via hash functions $h : [n] \to [w]$, $r : [n] \to \{-1, 1\}$



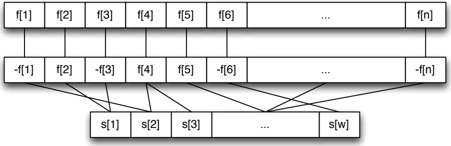
- *Update:* For each increment of f_i, $s_{h_i} \leftarrow s_{h_i} + r_i$. Hence,

$$s_k = \sum_{j : h_j = k} f_j r_j \qquad \text{e.g., } s_3 = f_6 - f_7 - f_{13}$$

- *Query:* Use $\tilde{f}_i = s_{h_i} r_i$ to estimate f_i.
- *Lemma:* $\mathbb{E}\left[\tilde{f}_i\right] = f_i$ and $\mathbb{V}\left[\tilde{f}_i\right] \leq F_2/w$

Count-Sketch: Count-Min with a Twist

- Maintain $s \in \mathbb{N}^w$ via hash functions $h : [n] \to [w]$, $r : [n] \to \{-1, 1\}$



- *Update:* For each increment of f_i, $s_{h_i} \leftarrow s_{h_i} + r_i$. Hence,

$$s_k = \sum_{j:h_j=k} f_j r_j \qquad \text{e.g., } s_3 = f_6 - f_7 - f_{13}$$

- *Query:* Use $\tilde{f}_i = s_{h_i} r_i$ to estimate f_i.
- *Lemma:* $\mathbb{E}\left[\tilde{f}_i\right] = f_i$ and $\mathbb{V}\left[\tilde{f}_i\right] \leq F_2/w$
- *Thm:* Let $w = O(1/\epsilon^2)$. Repeating $O(\lg \delta^{-1})$ in parallel and taking the median estimate ensures

$$\mathbb{P}\left[f_i - \epsilon\sqrt{F_2} \leq \tilde{f}_i \leq f_i + \epsilon\sqrt{F_2}\right] \geq 1 - \delta\ .$$

Proof of Lemma

- Define $\mathcal{E}$ by $\tilde{f}_i = f_i + \mathcal{E} r_i$ and so

$$\mathcal{E} = \sum_{j \neq i : h_i = h_j} f_j r_j$$

Proof of Lemma

- Define $\mathcal{E}$ by $\tilde{f}_i = f_i + \mathcal{E} r_i$ and so

$$\mathcal{E} = \sum_{j \neq i : h_i = h_j} f_j r_j$$

- *Expectation:* Since $\mathbb{E}[r_j] = 0$,

$$\mathbb{E}[\mathcal{E}] = \sum_{j \neq i : h_i = h_j} f_j \mathbb{E}[r_j] = 0$$

Proof of Lemma

- Define $\mathcal{E}$ by $\tilde{f}_i = f_i + \mathcal{E} r_i$ and so

$$\mathcal{E} = \sum_{j \neq i : h_i = h_j} f_j r_j$$

- *Expectation:* Since $\mathbb{E}[r_j] = 0$,

$$\mathbb{E}[\mathcal{E}] = \sum_{j \neq i : h_i = h_j} f_j \mathbb{E}[r_j] = 0$$

- *Variance:* Similarly,

$$\mathbb{V}[\mathcal{E}]$$

Proof of Lemma

- Define $\mathcal{E}$ by $\tilde{f}_i = f_i + \mathcal{E} r_i$ and so

$$\mathcal{E} = \sum_{j \neq i : h_i = h_j} f_j r_j$$

- *Expectation:* Since $\mathbb{E}[r_j] = 0$,

$$\mathbb{E}[\mathcal{E}] = \sum_{j \neq i : h_i = h_j} f_j \mathbb{E}[r_j] = 0$$

- *Variance:* Similarly,

$$\mathbb{V}[\mathcal{E}] \leq \mathbb{E}\left[\Big(\sum_{j \neq i : h_i = h_j} f_j r_j \Big)^2 \right]$$

Proof of Lemma

- Define $\mathcal{E}$ by $\tilde{f}_i = f_i + \mathcal{E} r_i$ and so

$$\mathcal{E} = \sum_{j \neq i : h_i = h_j} f_j r_j$$

- *Expectation:* Since $\mathbb{E}[r_j] = 0$,

$$\mathbb{E}[\mathcal{E}] = \sum_{j \neq i : h_i = h_j} f_j \mathbb{E}[r_j] = 0$$

- *Variance:* Similarly,

$$\mathbb{V}[\mathcal{E}] \le \mathbb{E}\left[(\sum_{j \neq i : h_i = h_j} f_j r_j)^2 \right] = \sum_{\substack{j,k \neq i \\ h_i = h_j = h_k}} f_j f_k \mathbb{E}[r_j r_k] \mathbb{P}[h_i = h_j = h_k]$$

Proof of Lemma

- Define $\mathcal{E}$ by $\tilde{f}_i = f_i + \mathcal{E} r_i$ and so

$$\mathcal{E} = \sum_{j \neq i : h_i = h_j} f_j r_j$$

- *Expectation:* Since $\mathbb{E}[r_j] = 0$,

$$\mathbb{E}[\mathcal{E}] = \sum_{j \neq i : h_i = h_j} f_j \mathbb{E}[r_j] = 0$$

- *Variance:* Similarly,

$$\begin{aligned} \mathbb{V}[\mathcal{E}] \leq \mathbb{E}\left[\left(\sum_{j \neq i : h_i = h_j} f_j r_j \right)^2 \right] &= \sum_{\substack{j,k \neq i \\ h_i = h_j = h_k}} f_j f_k \mathbb{E}[r_j r_k] \mathbb{P}[h_i = h_j = h_k] \\ &= \sum_{j \neq i : h_i = h_j} f_j^2 \mathbb{P}[h_i = h_j] \leq F_2 / w \end{aligned}$$

Outline

ℓ_p Sampling

ℓ_p Sampling

- *ℓ_p Sampling:* Return random values $I \in [n]$ and $R \in \mathbb{R}$ where

$$\mathbb{P}[I = i] = (1 \pm \epsilon)\frac{|f_i|^p}{F_p} \quad \text{and} \quad R = (1 \pm \epsilon) f_i$$

ℓ_p Sampling

- *ℓ_p Sampling:* Return random values $I \in [n]$ and $R \in \mathbb{R}$ where

$$\mathbb{P}\left[I = i\right] = (1 \pm \epsilon)\frac{|f_i|^p}{F_p} \quad \text{and} \quad R = (1 \pm \epsilon) f_i$$

- *Applications:*
  - Will use ℓ_2 sampling to get optimal algorithm for F_k, $k > 2$.
  - Will use ℓ_0 sampling for processing graph streams.
  - Many other stream problems can be solved via ℓ_p sampling, e.g., duplicate finding, triangle counting, entropy estimation.

ℓ_p Sampling

- *ℓ_p Sampling:* Return random values $I \in [n]$ and $R \in \mathbb{R}$ where

$$\mathbb{P}[I = i] = (1 \pm \epsilon)\frac{|f_i|^p}{F_p} \quad \text{and} \quad R = (1 \pm \epsilon) f_i$$

- *Applications:*
  - Will use ℓ_2 sampling to get optimal algorithm for F_k, $k > 2$.
  - Will use ℓ_0 sampling for processing graph streams.
  - Many other stream problems can be solved via ℓ_p sampling, e.g., duplicate finding, triangle counting, entropy estimation.
- Let's see algorithm for $p = 2 \ldots$

ℓ_2 Sampling Algorithm

ℓ_2 Sampling Algorithm

- Weight f_i by $\gamma_i = \sqrt{1/u_i}$ where $u_i \in_R [0,1]$ to form vector g:

$$\begin{aligned} f &= (f_1, f_2, \ldots, f_n) \\ g &= (g_1, g_2, \ldots, g_n) \quad \text{where } g_i = \gamma_i f_i \end{aligned}$$

ℓ_2 Sampling Algorithm

- Weight f_i by $\gamma_i = \sqrt{1/u_i}$ where $u_i \in_R [0,1]$ to form vector g:

$$\begin{aligned} f &= (f_1, f_2, \ldots, f_n) \\ g &= (g_1, g_2, \ldots, g_n) \quad \text{where } g_i = \gamma_i f_i \end{aligned}$$

- Return (i, f_i) if $g_i^2 \geq t := F_2(f)/\epsilon$

ℓ_2 Sampling Algorithm

- Weight f_i by $\gamma_i = \sqrt{1/u_i}$ where $u_i \in_R [0,1]$ to form vector g:

$$\begin{aligned} f &= (f_1, f_2, \ldots, f_n) \\ g &= (g_1, g_2, \ldots, g_n) \quad \text{where } g_i = \gamma_i f_i \end{aligned}$$

- Return (i, f_i) if $g_i^2 \geq t := F_2(f)/\epsilon$
- Probability (i, f_i) is returned:

$$\mathbb{P}\left[g_i^2 \geq t\right]$$

ℓ_2 Sampling Algorithm

- Weight f_i by $\gamma_i = \sqrt{1/u_i}$ where $u_i \in_R [0,1]$ to form vector g:

$$\begin{aligned} f &= (f_1, f_2, \ldots, f_n) \\ g &= (g_1, g_2, \ldots, g_n) \quad \text{where } g_i = \gamma_i f_i \end{aligned}$$

- Return (i, f_i) if $g_i^2 \geq t := F_2(f)/\epsilon$
- Probability (i, f_i) is returned:

$$\mathbb{P}\left[g_i^2 \geq t\right] = \mathbb{P}\left[u_i \leq f_i^2/t\right] = f_i^2/t$$

ℓ_2 Sampling Algorithm

- Weight f_i by $\gamma_i = \sqrt{1/u_i}$ where $u_i \in_R [0,1]$ to form vector g:

$$\begin{aligned} f &= (f_1, f_2, \ldots, f_n) \\ g &= (g_1, g_2, \ldots, g_n) \quad \text{where } g_i = \gamma_i f_i \end{aligned}$$

- Return (i, f_i) if $g_i^2 \geq t := F_2(f)/\epsilon$
- Probability (i, f_i) is returned:

$$\mathbb{P}\left[g_i^2 \geq t\right] = \mathbb{P}\left[u_i \leq f_i^2/t\right] = f_i^2/t$$

- Probability some value is returned is $\sum_i f_i^2/t = \epsilon$ so repeating $O(\epsilon^{-1} \log \delta^{-1})$ ensures a value is returned with probability $1 - \delta$.

ℓ_2 Sampling Algorithm

- Weight f_i by $\gamma_i = \sqrt{1/u_i}$ where $u_i \in_R [0,1]$ to form vector g:

$$\begin{aligned} f &= (f_1, f_2, \ldots, f_n) \\ g &= (g_1, g_2, \ldots, g_n) \quad \text{where } g_i = \gamma_i f_i \end{aligned}$$

- Return (i, f_i) if $g_i^2 \geq t := F_2(f)/\epsilon$
- Probability (i, f_i) is returned:

$$\mathbb{P}\left[g_i^2 \geq t\right] = \mathbb{P}\left[u_i \leq f_i^2/t\right] = f_i^2/t$$

- Probability some value is returned is $\sum_i f_i^2/t = \epsilon$ so repeating $O(\epsilon^{-1} \log \delta^{-1})$ ensures a value is returned with probability $1 - \delta$.
- *Lemma:* Using a Count-Sketch of size $O(\epsilon^{-1} \log^2 n)$ ensures a $(1 \pm \epsilon)$ approximation of any g_i that passes the threshold.

Proof of Lemma

Proof of Lemma

- *Exercise:* $\mathbb{P}\left[F_2(g)/F_2(f) \leq c \log n\right] \geq 99/100$ for some large $c > 0$ so we'll condition on this event.

Proof of Lemma

- *Exercise:* $\mathbb{P}\left[F_2(g)/F_2(f) \leq c \log n\right] \geq 99/100$ for some large $c > 0$ so we'll condition on this event.
- Set $w = 9c\epsilon^{-1} \log n$. Count-Sketch in $O(w \log^2 n)$ space ensures

$$\tilde{g}_i = g_i \pm \sqrt{F_2(g)/w}$$

Proof of Lemma

- *Exercise:* $\mathbb{P}\left[F_2(g)/F_2(f) \le c \log n\right] \ge 99/100$ for some large $c > 0$ so we'll condition on this event.
- Set $w = 9c\epsilon^{-1} \log n$. Count-Sketch in $O(w \log^2 n)$ space ensures

$$\tilde{g}_i = g_i \pm \sqrt{F_2(g)/w}$$

- Then $\tilde{g}_i^2 \ge F_2(f)/\epsilon$ implies

$$\sqrt{F_2(g)/w} \le \sqrt{F_2(f)/(9\epsilon^{-1})} \le \sqrt{\epsilon \tilde{g}_i^2/(9\epsilon^{-1})} = \epsilon \tilde{g}_i/3$$

  and hence $\tilde{g}_i^2 = (1 \pm \epsilon/3)^2 g_i^2 = (1 \pm \epsilon) g_i^2$ as required.

Proof of Lemma

- *Exercise:* $\mathbb{P}\left[F_2(g)/F_2(f) \leq c \log n\right] \geq 99/100$ for some large $c > 0$ so we'll condition on this event.
- Set $w = 9c\epsilon^{-1} \log n$. Count-Sketch in $O(w \log^2 n)$ space ensures

$$\tilde{g}_i = g_i \pm \sqrt{F_2(g)/w}$$

- Then $\tilde{g}_i^2 \geq F_2(f)/\epsilon$ implies

$$\sqrt{F_2(g)/w} \leq \sqrt{F_2(f)/(9\epsilon^{-1})} \leq \sqrt{\epsilon \tilde{g}_i^2/(9\epsilon^{-1})} = \epsilon \tilde{g}_i/3$$

  and hence $\tilde{g}_i^2 = (1 \pm \epsilon/3)^2 g_i^2 = (1 \pm \epsilon) g_i^2$ as required.
- *Under-the-rug:* Need to ensure that conditioning doesn't affect sampling probability too much.

F_k Revisited

- Earlier we used $O(n^{1-1/k})$ space to approximate $F_k = \sum_i |f_i|^k$.

F_k Revisited

- Earlier we used $O(n^{1-1/k})$ space to approximate $F_k = \sum_i |f_i|^k$.
- *Algorithm:* Let (I, R) be an $(1+\gamma)$-approximate ℓ_2 sample. Return

$$T = \tilde{F}_2 R^{k-2} \qquad \text{where } \tilde{F}_2 \text{ is a } (1 \pm \gamma) \text{ approximation for } F_2$$

F_k Revisited

- Earlier we used $O(n^{1-1/k})$ space to approximate $F_k = \sum_i |f_i|^k$.
- *Algorithm:* Let (I, R) be an $(1+\gamma)$-approximate ℓ_2 sample. Return

$$T = \tilde{F}_2 R^{k-2} \qquad \text{where } \tilde{F}_2 \text{ is a } (1 \pm \gamma) \text{ approximation for } F_2$$

- *Expectation:* Setting $\gamma = \epsilon/(4k)$,

$$\mathbb{E}\left[T\right]$$

F_k Revisited

- Earlier we used $O(n^{1-1/k})$ space to approximate $F_k = \sum_i |f_i|^k$.
- *Algorithm:* Let (I, R) be an $(1+\gamma)$-approximate ℓ_2 sample. Return

$$T = \tilde{F}_2 R^{k-2} \qquad \text{where } \tilde{F}_2 \text{ is a } (1 \pm \gamma) \text{ approximation for } F_2$$

- *Expectation:* Setting $\gamma = \epsilon/(4k)$,

$$\mathbb{E}[T] = \tilde{F}_2 \sum \mathbb{P}[I = i]\,((1\pm\gamma) f_i)^{k-2}$$

F_k Revisited

- Earlier we used $O(n^{1-1/k})$ space to approximate $F_k = \sum_i |f_i|^k$.
- *Algorithm:* Let (I, R) be an $(1+\gamma)$-approximate ℓ_2 sample. Return

$$T = \tilde{F}_2 R^{k-2} \qquad \text{where } \tilde{F}_2 \text{ is a } (1 \pm \gamma) \text{ approximation for } F_2$$

- *Expectation:* Setting $\gamma = \epsilon/(4k)$,

$$\mathbb{E}[T] = \tilde{F}_2 \sum \mathbb{P}[I = i]((1\pm\gamma)f_i)^{k-2} = (1\pm\gamma)^k F_2 \sum \frac{f_i^2}{F_2} f_i^{k-2}$$

F_k Revisited

- Earlier we used $O(n^{1-1/k})$ space to approximate $F_k = \sum_i |f_i|^k$.
- *Algorithm:* Let (I, R) be an $(1+\gamma)$-approximate ℓ_2 sample. Return

$$T = \tilde{F}_2 R^{k-2} \qquad \text{where } \tilde{F}_2 \text{ is a } (1 \pm \gamma) \text{ approximation for } F_2$$

- *Expectation:* Setting $\gamma = \epsilon/(4k)$,

$$\mathbb{E}\left[T\right] = \tilde{F}_2 \sum \mathbb{P}\left[I = i\right]((1\pm\gamma)f_i)^{k-2} = (1\pm\gamma)^k F_2 \sum \frac{f_i^2}{F_2} f_i^{k-2} = (1\pm\frac{\epsilon}{2})F_k$$

F_k Revisited

- Earlier we used $O(n^{1-1/k})$ space to approximate $F_k = \sum_i |f_i|^k$.
- *Algorithm:* Let (I, R) be an $(1+\gamma)$-approximate ℓ_2 sample. Return

$$T = \tilde{F}_2 R^{k-2} \qquad \text{where } \tilde{F}_2 \text{ is a } (1 \pm \gamma) \text{ approximation for } F_2$$

- *Expectation:* Setting $\gamma = \epsilon/(4k)$,

$$\mathbb{E}[T] = \tilde{F}_2 \sum \mathbb{P}[I = i]((1 \pm \gamma) f_i)^{k-2} = (1 \pm \gamma)^k F_2 \sum \frac{f_i^2}{F_2} f_i^{k-2} = (1 \pm \frac{\epsilon}{2}) F_k$$

- *Range:* $0 \le T \le (1+\gamma) F_2 F_\infty^{k-2} = (1+\gamma) n^{1-2/k} F_k$.

F_k Revisited

- Earlier we used $O(n^{1-1/k})$ space to approximate $F_k = \sum_i |f_i|^k$.
- *Algorithm:* Let (I, R) be an $(1+\gamma)$-approximate ℓ_2 sample. Return

$$T = \tilde{F}_2 R^{k-2} \qquad \text{where } \tilde{F}_2 \text{ is a } (1 \pm \gamma) \text{ approximation for } F_2$$

- *Expectation:* Setting $\gamma = \epsilon/(4k)$,

$$\mathbb{E}[T] = \tilde{F}_2 \sum \mathbb{P}[I = i]((1 \pm \gamma) f_i)^{k-2} = (1 \pm \gamma)^k F_2 \sum \frac{f_i^2}{F_2} f_i^{k-2} = (1 \pm \frac{\epsilon}{2}) F_k$$

- *Range:* $0 \leq T \leq (1+\gamma) F_2 F_\infty^{k-2} = (1+\gamma) n^{1-2/k} F_k$.
- Averaging over $t = O(\epsilon^{-2} n^{1-2/k} \log \delta^{-1})$ parallel repetitions gives,

$$\mathbb{P}\left[|\tilde{F}_k - F_k| \geq \epsilon F_k\right] \leq \delta$$

F_k Revisited

- Earlier we used $O(n^{1-1/k})$ space to approximate $F_k = \sum_i |f_i|^k$.
- *Algorithm:* Let (I, R) be an $(1+\gamma)$-approximate ℓ_2 sample. Return

$$T = \tilde{F}_2 R^{k-2} \qquad \text{where } \tilde{F}_2 \text{ is a } (1 \pm \gamma) \text{ approximation for } F_2$$

- *Expectation:* Setting $\gamma = \epsilon/(4k)$,

$$\mathbb{E}[T] = \tilde{F}_2 \sum \mathbb{P}[I = i]((1\pm\gamma)f_i)^{k-2} = (1\pm\gamma)^k F_2 \sum \frac{f_i^2}{F_2} f_i^{k-2} = (1\pm\frac{\epsilon}{2})F_k$$

- *Range:* $0 \leq T \leq (1+\gamma)F_2 F_\infty^{k-2} = (1+\gamma)n^{1-2/k}F_k$.
- Averaging over $t = O(\epsilon^{-2}n^{1-2/k}\log\delta^{-1})$ parallel repetitions gives,

$$\mathbb{P}\left[|\tilde{F}_k - F_k| \geq \epsilon F_k\right] \leq \delta$$

- *Thm:* In $\tilde{O}(\epsilon^{-2}n^{1-2/k})$ space we can find a $(1\pm\epsilon)$ approximation for F_k with probability at least $1-\delta$.

Summary

- *Basic Sampling:* Can do basic sampling where i is selected with probability $\propto f_i$ but we can be much smarter via sketches.
- *Count-Min:* $f_i \leq \tilde{f}_i \leq f_i + \epsilon F_1$ in $O(\epsilon^{-1})$ space.

$$Z = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

- *Count-Sketch:* $f_i - \epsilon\sqrt{F_2} \leq \tilde{f}_i \leq f_i + \epsilon\sqrt{F_2}$ in $O(\epsilon^{-2})$ space.

$$Z = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ -1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

  Above sketches solve range-queries, quantiles, heavy hitters, ...
- *ℓ_p-Sampling:* Selecting i with probability $\propto f_i^p$ in $O(\epsilon^{-2})$ space.

$$Z = \begin{bmatrix} 0 & \gamma_2 & 0 & 0 & 0 & -\gamma_6 \\ 0 & 0 & 0 & -\gamma_4 & 0 & 0 \\ -\gamma_1 & 0 & \gamma_3 & 0 & \gamma_5 & 0 \end{bmatrix}$$